Numerical HW 8 - CSCI 3656 - 001
Kai Handelman
10/30/21

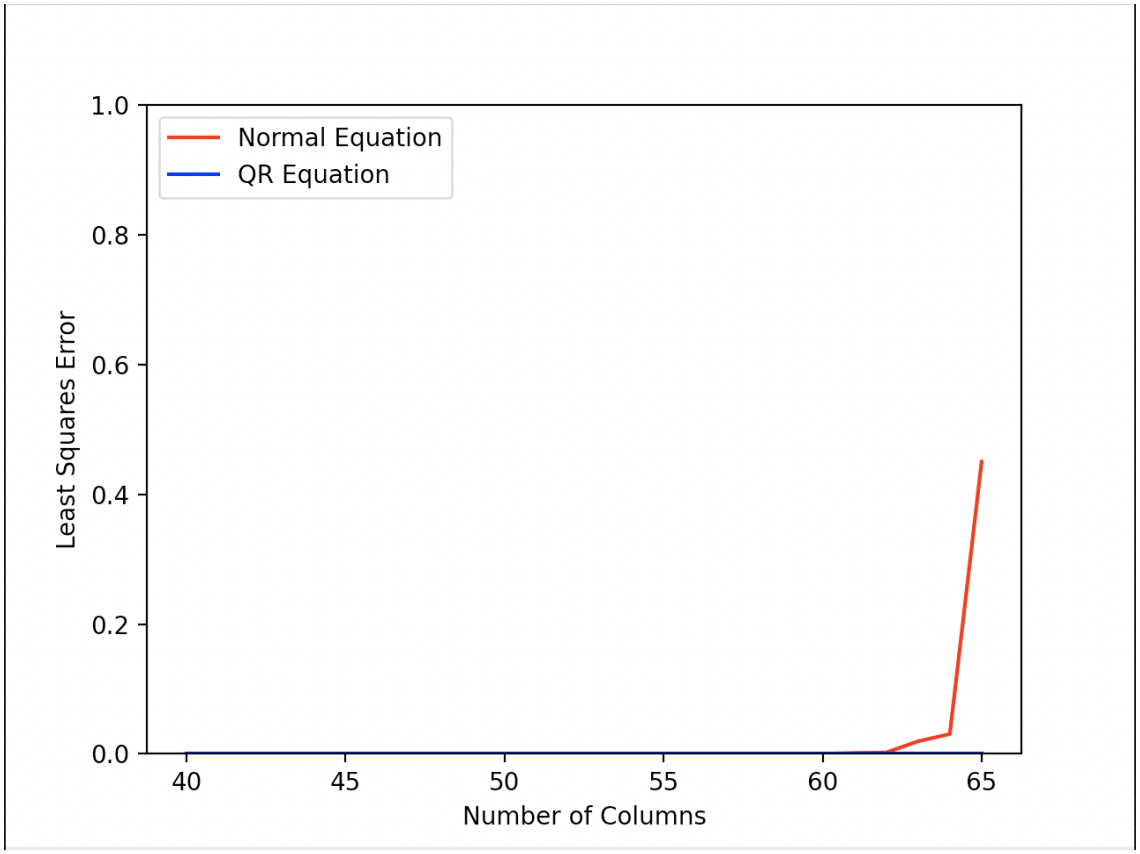
Size, Rank, and Condition Number for each A_k:

K	Size	Rank	Condition Number
40	101x40	40	74.8767
41	101x41	41	103.8
42	101x42	42	152.286
43	101x43	43	217.56
44	101x44	44	328.892
45	101x45	45	483.781
46	101x46	46	753.046
47	101x47	47	1140.07
48	101x48	48	1826.79
49	101x49	49	2846.42
50	101x50	50	4695.09
51	101x51	51	7530.55
52	101x52	52	12789.4
53	101x53	53	21122.7
54	101x54	54	36949.5
55	101x55	55	62868.3
56	101x56	56	113329
57	101x57	57	198771
58	101x58	58	369476
59	101x59	59	668493
60	101x60	60	1.28227e+06
61	101x61	61	2.3953e+06
62	101x62	62	4.74546e+06
63	101x63	63	9.16153e+06
64	101x64	64	1.87657e+07
65	101x65	65	3.74863e+07

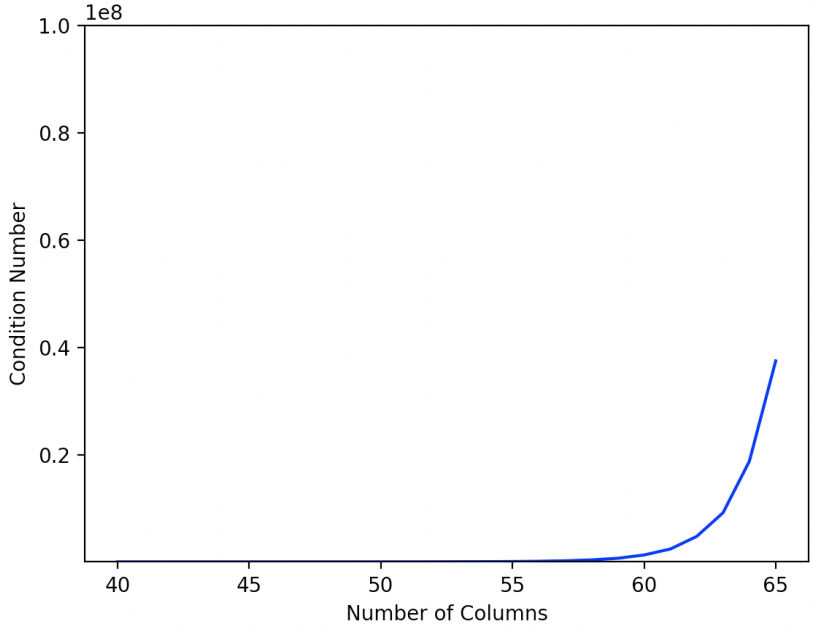
Average Error over all the b (Separated by Column count):

K	NE Error Average	QR Error Average
40	4.51678e-13	1.81167e-14
41	1.79886e-12	1.09264e-14
42	1.97752e-12	2.62669e-14
43	7.76176e-12	2.9419e-14
44	8.87174e-12	3.45993e-14
45	3.83907e-11	1.49622e-14
46	4.72765e-11	9.02397e-14
47	2.09924e-10	1.16061e-13
48	2.74369e-10	1.0012e-13
49	1.29175e-09	7.22438e-13
50	1.81123e-09	2.9778e-13
51	8.57298e-09	4.93581e-13
52	8.35087e-09	4.16352e-13
53	6.43309e-08	1.89088e-12
54	6.59757e-08	3.12539e-12
55	6.42026e-07	1.39506e-11
56	1.06198e-06	4.06765e-12
57	6.43513e-06	2.26792e-11
58	1.12249e-05	3.84889e-11
59	7.35855e-05	8.129e-11
60	0.000135114	1.7492e-10
61	0.00134051	7.84489e-11
62	0.00200407	6.07811e-10
63	0.0189368	9.33393e-10
64	0.0298006	4.79669e-08
65	0.485797	5.70672e-08

Graphs for the Error Comparision (Couldn't Get the y-axis to work properly):



Graphs for Condition Number Vs. Column Number (Couldn't Get the y-axis to work properly):



Reflection:

1) Though a bit hard to visualize on the provided graph, the table printed above clearly shows the QR equation was able to contain its error much better as the number of columns increased compared to the Normal Equation.
2) There seems like there is a correlation between condition number and error for both QR and Normal Equations. Both errors and the condition number increase as the number of columns of the matrix increases.
3) Since QR's error increased at a much slower pace than the Normal Equation, for ill-conditioned matrix, it'd be better to use the QR method over the Normal Equation Method.

Code:

```
import numpy as np
import numpy.linalg as ln
import numpy.random as rn
from tabulate import tabulate
import matplotlib.pyplot as plt

def readFile(fileName):
    returnContent = []
    with open(fileName,'r') as f:
        contents = f.readlines()
        for i in contents:
            returnContent.append(i.split(","))
        for i in returnContent:
            i[len(i)-1] = i[len(i)-1].replace("\n","")
            for j in range(len(i)):
                i[j] = float(i[j])

        return returnContent

def getTranspose(matrix):
    tMatrix = [[0 for j in range(len(matrix))] for i in range(len(matrix[0]))]
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            tMatrix[j][i] = matrix[i][j]
    return tMatrix

def normEq(a,b):
    aTa = np.dot(getTranspose(a),a)
    atB = np.dot(getTranspose(a),b)
    try:
        r = ln.cholesky(aTa)
    except:
        print("Not positive definite")
        return None
    y = ln.solve(r,atB)
    return ln.solve(getTranspose(r),y)

def backSub(matrix,right):
```

```
    x = np.zeros_like(right,float)
    for i in range(len(matrix)-1,-1,-1):
        temp = right[i][0]
        for j in range(len(matrix[i])-1, i, -1):
            temp = temp - matrix[i][j] * x[j]
        x[i] = temp / (matrix[i][i])
    return x

def thinQR(a,b):
    q,r = ln.qr(a,mode='reduced')
    nB = np.dot(getTranspose(q), b)
    x = backSub(r,nB)
    return x

def generateRight(m,Count):
    bs = []
    for i in range(Count):
        bs.append(rn.rand(m,1))
    return bs

def problemTwo(tempA,b):
    # Part A
    trueX = []

    errNormalEq = []
    errQREq = []
    for i in tempA:
        x = ln.lstsq(i,b,rcond = -1)[0]
        trueX.append(x)

        ne_X = normEq(i,b)
        errNormalEq.append(ln.norm(ne_X-x)/ln.norm(x))

        ne_X = thinQR(i,b)
        errQREq.append(ln.norm(ne_X-x)/ln.norm(x))

    return  errNormalEq,errQREq

def main():
```

```
    matrix = np.array(readFile("mat1-2.txt"))
    # Question One
    kmin = 40
    kmax = 65
    kArrry = []
    pOneData = [["K","Size","Rank","Condition Number"]]
    kCon = []
    for i in range(kmin,kmax+1):
        tempMatrix = matrix[:,0:i]
        kArrry.append(tempMatrix)
        pOneData.append([i,str(len(tempMatrix))  + "x" + str(len(tempMatrix[0])),
ln.matrix_rank(tempMatrix), ln.cond(tempMatrix)])
        kCon.append(ln.cond(tempMatrix))
    print(tabulate(pOneData,headers="firstrow",tablefmt='grid')+ "\n")

    # Question Two
    randomB = generateRight(len(matrix[0]),100)
    ne = []
    qr = []

    for i in randomB:
        neTemp,qrTemp = problemTwo(kArrry,i)
        ne.append(neTemp)
        qr.append(qrTemp)

    #Question Three
    avgErr = []
    neData = []
    qrData = []
    for i in range(26):
        tempSumQR = 0
        tempSumNE = 0
        for j in range(len(randomB)):
            tempSumQR = tempSumQR + qr[j][i]
            tempSumNE = tempSumNE + ne[j][i]
        avgErr.append([i+kmin,tempSumNE/26,tempSumQR/26])
        neData.append(tempSumNE/26)
        qrData.append(tempSumQR/26)
```

```
    pThreeData = [["K","NE Error Average","QR Error Average"]]
    pThreeData = pThreeData + avgErr
    print(tabulate(pThreeData,headers="firstrow",tablefmt='grid')+ "\n")



    # GRAPH MAKING
    xData = list(range(kmin,kmax+1))
    plt.xlabel("Number of Columns")
    plt.ylabel("Least Squares Error")
    plt.plot(xData,neData,'r',label="Normal Equation")
    plt.plot(xData,qrData,'b',label="QR Equation")
    plt.ylim([10**(-15), 1])
    plt.legend(loc="upper left")
    plt.show()

    plt.plot(xData,kCon,'b')
    plt.xlabel("Number of Columns")
    plt.ylabel("Condition Number")
    plt.ylim([np.power(10,2), np.power(10,8)])
    plt.show()


def test():
    a = np.array([[1,2,3],[2,2,2],[5,5,5]])
    print(a[0:2,0:2])

# test()
main()
```